Metodologie di testing nell'ambiente Android

A cura di Lorenzo Rigoni

Cosa sono i test di un'app?

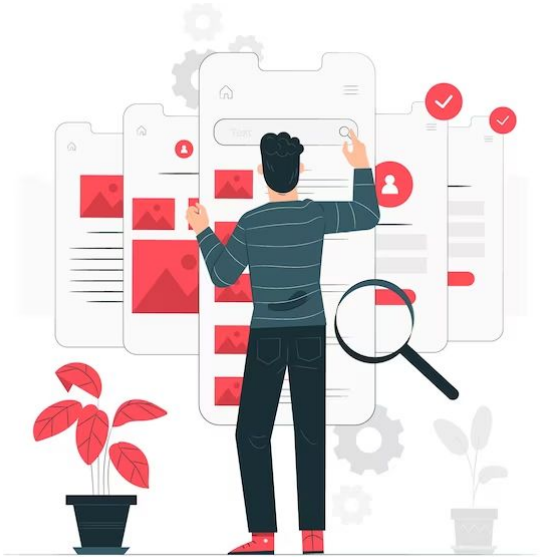
- parte integrante del processo di sviluppo
- verificare la correttezza, il comportamento funzionale e l'usabilità

Perché sono importanti i test?

- Garantire qualità e affidabilità
- Migliorare l'esperienza utente e la reputazione dell'applicazione

Tipi di test:

1. **Test funzionale**: l'app svolge il suo compito?
2. **Test del rendimento**: l'app è veloce ed efficiente?
3. **Test di accessibilità**: l'app è accessibile da tutti?
4. **Test di compatibilità**: l'app funziona su ogni tipo di dispositivo?



E in Android?

Diversi tipi di testing:

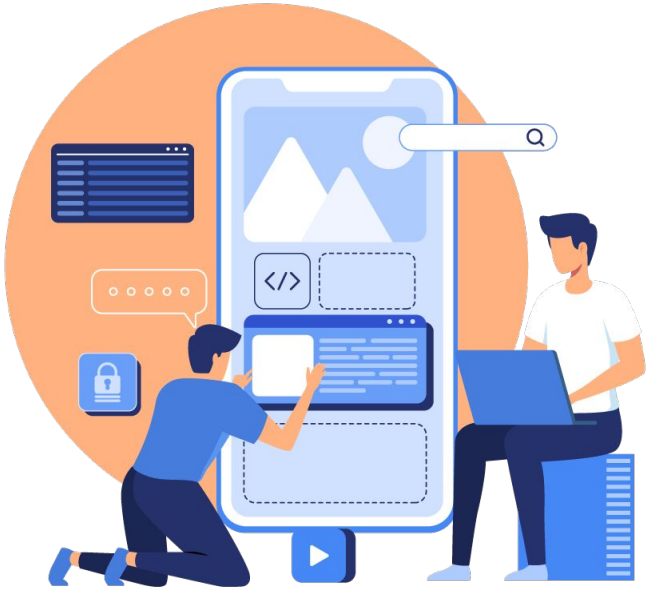
- **Unit testing**
- **Integration testing**
- **UI testing**

In generale:

- **Test strumentali**: dispositivo Android (fisico o emulato)
- **Test locali**: dispositivo di sviluppo o server

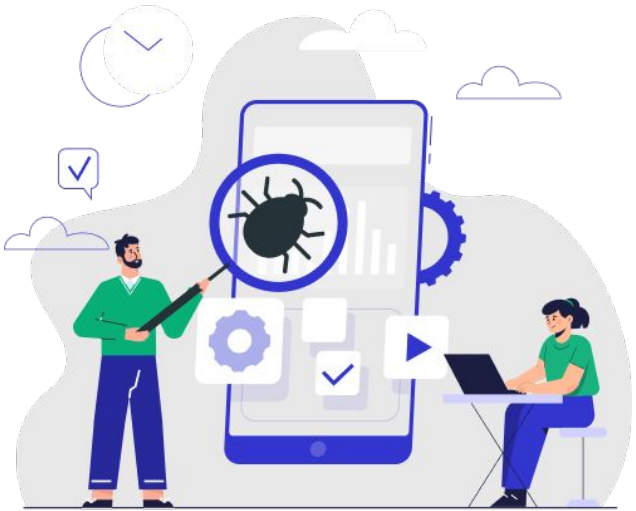
Strategie e test irregolari

- Impossibile testare ogni riga di codice
- Equilibrio tra fedeltà di un test, velocità e affidabilità
- Test di fedeltà = somiglianza dell'ambiente con un dispositivo reale
- **Double test**
- Errori anche nelle esecuzioni di test progettate ed implementate correttamente
- Test non passato al 100% = **test irregolare**



Struttura testabile

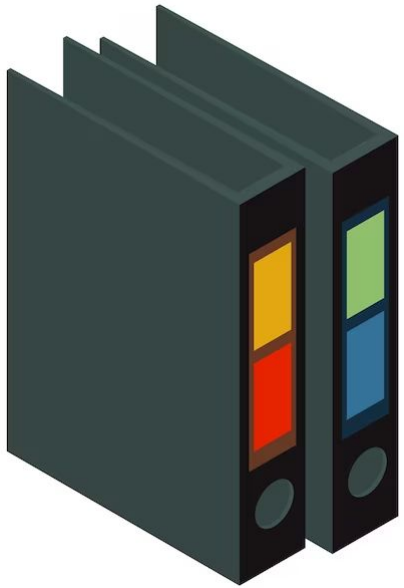
- Il codice segue una struttura facile da testare
- Vantaggi come migliore leggibilità, manutenibilità, scalabilità e riusabilità
- Architettura non testabile = test più grandi, più lenti e più irregolari



Directory per i test

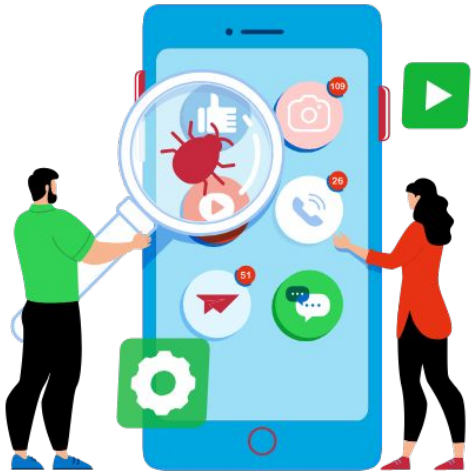
In un progetto Android, possiamo trovare due directory che contengono i test:

- **androidTest**: test strumentali
- **test**: test locali



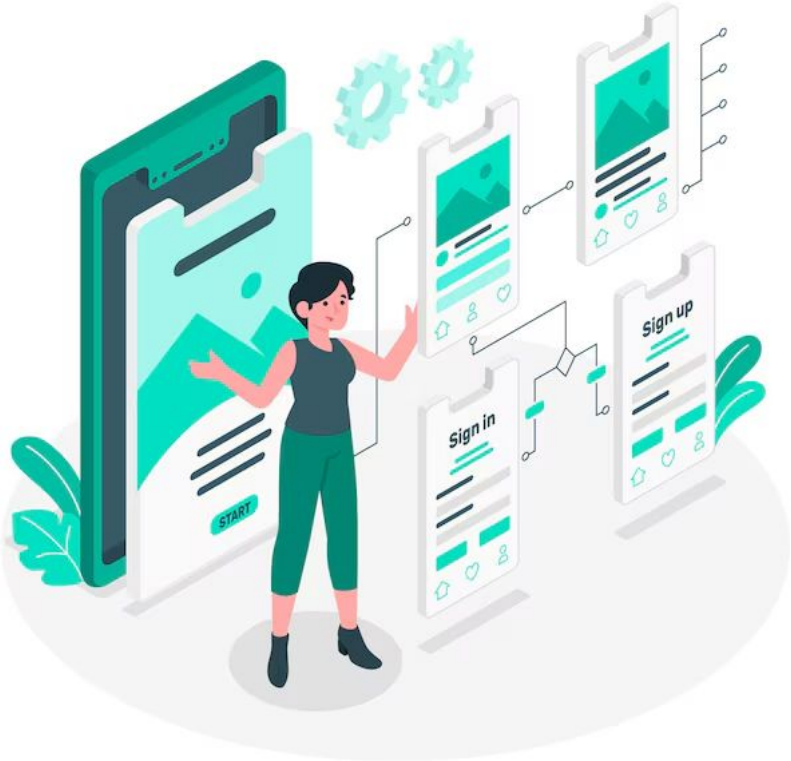
Unit testing

- **Unit test per ViewModels**
- **Unit test per i dati** (repository e database)
- **Unit test per classi** (manipolazione di stringhe e calcoli matematici)
- **Unit test per casi limite** (scenari rari)



UI testing

- **Screen UI tests** (interazioni critiche degli utenti)
- **User flow tests** (navigazione dell'utente all'interno dell'app)
- **Monkey test** (programma che genera flussi pseudo-casuali)



Test doubles

Nella progettazione della strategia di test, tre aspetti correlati:

- **Ambito**: quanta parte di codice viene usata dal test?
- **Velocità**: quanto è veloce il test?
- **Fedeltà**: quanto è realistico il test?

Diventa utile ***isolare*** gli elementi. Però, un elemento potrebbe essere dipendente da un altro.

Pratica comune: creare un ***test object*** o ***test double***

Test doubles

Tipi di test doubles:

- **Fake** (implementazione funzionante ma utile solo per i test, non per la produzione)
- **Mock** (svolge il compito assegnato ed ha un controllo prestabilito sulle interazioni)
- **Stub** (svolge il compito assegnato ma non ha controlli sulle interazioni)
- **Dummy** (non viene usato finché non viene passato come parametro)
- **Spy** (wrapper su oggetti reali ma molto complicati. Da preferire i fake o i mock)

Conclusioni

In questa presentazione sono state approfondite le metodologie di testing in Android, evidenziandone l'importanza nei progetti e consigliandone fortemente l'uso.

Grazie per l'attenzione